Trash to Treasure: Harvesting OOD Data with Cross-Modal Matching for Open-Set Semi-Supervised Learning
– *Supplementary Material*

Junkai Huang[1*] Chaowei Fang[2*] Weikai Chen[3] Zhenhua Chai[4]
Xiaolin Wei[4] Pengxu Wei[1] Liang Lin[1] Guanbin Li[1†]
[1]Sun Yat-sen University [2]Xidian University [3]Tencent America [4]Meituan

1. Supplementary Ablation Study

1.1. Incorporation with Other SSL Methods

Various variants of our method can be devised via replacing UDA with other SSL methods. The performance of integrating existing SSL algorithms including self-labeling [4], mean teacher [6], and MixMatch [1] into our proposed framework is presented in Table 1. Here, CIFAR-10 and TIN is regarded as the ID and OOD dataset respectively and 1000 labels are used during training. Our method can consistently improve the three SSL algorithms. This demonstrates our method can be easily integrated with various SSL algorithms.

Methods	Self-labeling [4]	Mean Teacher [6]	MixMatch [1]
Baseline	62.60	72.39	88.03
Ours	**82.51**	**84.74**	**92.17**

Table 1: Performance of incorporating our proposed method with different SSL methods.

1.2. Training with Different Sampling Strategies

As shown in Table 2, we validate the efficacy of our devised sampling strategy for training the cross-modal matching branch on various ID and OOD datasets. 50 labels are used in every category of all datasets during training. We find that using the hard sample only contributes to better OOD detection performance than using the simple sample only, since discriminating hard samples is more challenging and requires the learned features to be more expressive, compared to learning the simple samples. The combination of hard and simple samples brings a significant performance improvement in OOD detection, in contrast to only using the hard/simple training sample.

ID	OOD	hard only	simple only	hard + simple
Animals-10	TIN	90.32	88.79	**93.51**
CIFAR-ID-50	TIN	83.98	77.23	**99.85**
CIFAR-ID-50	CIFAR-50	70.07	65.81	**74.13**
TIN-ID-50	TIN-150	63.78	61.89	**65.67**

Table 2: The OOD detection performance of using different sampling strategies when training the cross-modal matching branch. AUROC(%) is used as the metric for measuring OOD detection performance.

1.3. Using Different Network Backbones

To verify whether our method is sensitive to the network backbone, we implement variants of our method via adopting VGG13 [5], ResNet18 [2], or PreAct-ResNet18 [3] as the visual feature extraction backbone. The comparisons between our method and the baseline method, UDA [7], are presented in Table 3. In all experiments, we use CIFAR-10 as the in-distribution dataset, and regard TIN/LSUN as the out-of-distribution dataset. 25 labeled samples are provided for each class. Our method can outperform UDA significantly on all network backbones.

Backbone	TIN		LSUN	
	UDA	Ours	UDA	Ours
VGG13 [5]	88.04	**90.43**	87.42	**90.31**
ResNet18 [2]	88.59	**90.72**	88.70	**90.04**
PreAct-ResNet18 [3]	87.73	**90.61**	87.28	**90.56**

Table 3: Performance of incorporating our proposed method with different network backbones.

*Equal contribution.
†Corresponding author.

1.4. Influence of Length of OOD Filtering Cycle

The crossmodal matching head is used to periodically update the unlabeled training data. We test extensive OOD filtering cycle lengths $\{10^4, 2\times10^4, 4\times10^4, 10^5, 2\times10^5\}$ to analyze the efficacy of our method under different data update frequencies. In all experiments, we use the same ID and OOD dataset settings as in Section 1.3. The experimental results are shown in Table 4. We can see that shortening the filtering period to as low as 2×10^4 iterations, namely updating the unlabeled training data more frequently, gives rise to continuous improvements in the accuracy of our method. However, the overly frequent update makes the model formidable to adapt to data variation, resulting in performance decay.

OOD Dataset	Cycle Length (10^4)				
	1	2	4	10	20
TIN	90.26	**91.52**	89.34	86.93	85.66
LSUN	89.78	**91.13**	88.17	86.51	84.79

Table 4: Influence of Length of OOD Filtering Cycle.

1.5. Dimension of Class Embedding

In the cross-modal matching branch, we transform the one-hot vector of a class label into a d-dimensional class embedding vector. In order to test the sensitivity of different dimensions d to the performance of the algorithm, we set d to 64, 128, 256, and 512 respectively, and report the OOD detection performance under different class embedding dimensions in Table 5. Here, CIFAR-10 or CIFAR-ID-50 is selected as ID data, and TIN is regarded as OOD data. 250 and 2500 labeled images are for CIFAR-10 and CIFAR-ID-50, respectively. As shown in Table 5, our method performs well on the OOD detection for all dimension settings, indicating that our approach is not sensitive to the dimension of class embedding.

Dataset	Dimension			
	64	128	256	512
CIFAR-10	99.93	99.97	99.95	99.97
CIFAR-ID-50	99.89	99.85	99.86	99.91

Table 5: Dimension selection of class embedding. AUROC(%) is used as the metric for measuring OOD detection performance.

2. Supplementary Training Algorithms

The training process consists of two stages. In the first stage, we take a warming up training of the complete architecture with loss function $L = L_{ce} + L^l_{cm} + L_{rot}$. In the second stage, the cross-modal matching head is used to periodically update unlabeled data. To further improve the performance of the category prediction branch and the cross-modal matching branch, consistency constraint and entropy minimization are adopted, respectively. The final loss function is thus $L = L_{ce} + L^l_{cm} + L^u_{cm} + L_{rot} + L_{cc}$. The calculation functions of L_{ce} and L_{cc} are described in Algorithm 1. The loss function of the rotation prediction is presented in Algorithm 2. The calculation procedures of L^l_{cm} and L^u_{cm} are illustrated in Algorithm 3. We introduce the dataset updating process in Algorithm 4. The processes of the two training stages are summarized in Algorithm 5 and 6, respectively.

Algorithm 1: Functions of computing losses, L_{ce} and L_{cc}, for the category prediction task. $\mathbf{x}$, y, θ, and ω_c indicates the input image, the category label, parameters of the backbone and classification head, respectively.

Function `loss_ce` ($\mathbf{x}$, y, θ, ω_c) **:**
  $\mathbf{f} \leftarrow g_\theta(\mathbf{x})$; $\mathbf{p} \leftarrow h_{\omega_c}(\mathbf{f})$;
  return $-\ln(p[y])$, $\mathbf{f}$, $\mathbf{p}$;
Function `loss_cc` ($\mathbf{x}$, θ, ω_c) **:**
  $\mathbf{f} \leftarrow g_\theta(\mathbf{x})$; $\mathbf{p} \leftarrow h_{\omega_c}(\mathbf{f})$;
  Strongly augment $\mathbf{x}$, resulting to $\tilde{\mathbf{x}}$;
  $\tilde{\mathbf{p}} \leftarrow h_{\omega_c}(g_\theta(\tilde{\mathbf{x}}))$
  return $\sum_{j=1}^{K} p[j]\ln(\frac{p[j]}{\tilde{p}[j]})$, $\mathbf{f}$, $\mathbf{p}$;
Function `loss_ce_cc` ($\mathbf{x}$, y, θ, ω_c) **:**
  $\mathbf{f} \leftarrow g_\theta(\mathbf{x})$; $\mathbf{p} \leftarrow h_{\omega_c}(\mathbf{f})$;
  Strongly augment $\mathbf{x}$, resulting to $\tilde{\mathbf{x}}$;
  $\tilde{\mathbf{p}} \leftarrow h_{\omega_c}(g_\theta(\tilde{\mathbf{x}}))$
  return $-\ln(p[y])$, $\sum_{j=1}^{K} p[j]\ln(\frac{p[j]}{\tilde{p}[j]})$, $\mathbf{f}$, $\mathbf{p}$;

Algorithm 2: Function of computing loss for the rotation prediction task, L_{rot}. ω_r represents the parameters of the rotation prediction head.

Function `loss_rot` ($\mathbf{x}$, θ, ω_r) **:**
  $L \leftarrow 0$
  for $j \leftarrow 1$ **to** *4* **do**
    Rotate $\mathbf{x}_i$ by $(j-1)*90°$, resulting to $\mathbf{x}_{i,j}$;
    $\mathbf{q}_{i,j} \leftarrow h_{w_r}(g_\theta(\mathbf{x}_{i,j}))$;
    $L \leftarrow L - \ln(q_{i,j}[j])$;
  end
  return $L/4$.

Algorithm 3: Functions of calculating loss for the cross-modal matching task. $loss\_cm\_labeled$ and $loss\_cm\_unlabeled$ is used for calculating losses of labeled and unlabeled samples, L_{cm}^{l} and L_{cm}^{u}, respectively. $\mathbf{f}$, y, and $\mathbf{p}$ indicates the feature representation , category label , and the predicted probability vector of a training sample, respectively. ϕ and ω_m represent the parameters of the cross-modal matching branch.

Function `loss_cm_labeled`($\mathbf{f}$, y, $\mathbf{p}$, ϕ, ω_m)**:**
- $s \leftarrow h_{\omega_m}(\mathbf{f}, g_\phi(y))$;
- $\bar{y}^h \leftarrow \arg\max_{k \neq y} p[k]$;
- $\bar{s}^h \leftarrow h_{\omega_m}(\mathbf{f}, g_\phi(\bar{y}^h))$;
- $\bar{y}^s \leftarrow \text{rand}(\{k \in [1, K] \mid k \neq y;\ k \neq \bar{y}^h\})$;
- $\bar{s}^s \leftarrow h_{\omega_m}(\mathbf{f}, g_\phi(\bar{y}^s))$;
- **return** $-\ln(s) - \ln(1 - \bar{s}^h) - \ln(1 - \bar{s}^s)$;

Function `loss_cm_unlabeled`($\mathbf{f}$, $\mathbf{p}$, ϕ, ω_m)**:**
- $y \leftarrow \arg\max_k p[k]$;
- $s \leftarrow h_{\omega_m}(\mathbf{f}, g_\phi(y))$;
- $\bar{y} \leftarrow \text{rand}(\{k \in [1, K] \mid k \neq y\})$;
- $\bar{s} \leftarrow h_{\omega_m}(\mathbf{f}, g_\phi(\bar{y}))$;
- **return** $-s\ln(s) - (1-s)\ln(1-s) - \bar{s}\ln(\bar{s}) - (1-\bar{s})\ln(1-\bar{s})$;

Algorithm 4: Function of filtering samples. $\mathbb{D}$ represents a set of unlabeled images.

Function `sample_filtering`($\mathbb{D}$, θ, ϕ, ω_c, ω_m)**:**
- **foreach** $\mathbf{x}_i \in \mathbb{D}$ **do**
  - $\mathbf{f}_i \leftarrow g_\theta(\mathbf{x}_i)$; $\mathbf{p}_i \leftarrow h_{\omega_c}(\mathbf{f}_i)$;
  - $y_i \leftarrow \arg\max_k p_i[k]$; $s_i \leftarrow h_{\omega_m}(\mathbf{f}_i, g_\phi(y_i))$;
- **end**
- Obtain the Otsu thresholding τ from $\{s_i\}$;
- $\mathbb{D}' \leftarrow \emptyset$
- **foreach** $\mathbf{x}_i \in \mathbb{D}$ **do**
  - **if** $s_i > \tau$ **then**
    - Push $\mathbf{x}_i$ into $\mathbb{D}'$;
- **end**
- **return** $\mathbb{D}'$;

References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*, pages 630–645. Springer, 2016. 1

[4] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, 2013. 1

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[6] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, 2017. 1

[7] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc., 2020. 1

Algorithm 5: Algorithm for network warming up.

Input: Training dataset, $\left\{(\mathbf{x}_i^l, y_i^l)\right\}_{i=1}^N$, $\{\mathbf{x}_i^u\}_{i=1}^M$.
Output: Optimized network parameters, θ, ϕ, ω_c, ω_m, ω_r.
Randomly initialize network parameters.
repeat
  Fetch a batch of labeled samples $\mathbb{B}^l = \{(\mathbf{x}_i^l, y_i^l)\}_{i=1}^n$ and a batch of unlabeled samples $\mathbb{B}^u = \{\mathbf{x}_i^u\}_{i=1}^m$;
  $L_{sup} \leftarrow 0$; $L_{cm}^l \leftarrow 0$;
  for $i \leftarrow 1$ **to** n **do**
    $L_{ce}^i, \mathbf{f}_i^l, \mathbf{p}_i^l \leftarrow loss\_ce(\mathbf{x}_i^l, y_i^l, \theta, \omega_c)$;
    $L_{ce} \leftarrow L_{ce} + L_{ce}^i$;
    $L_{cm}^l \leftarrow L_{cm}^l + loss\_cm\_labeled(\mathbf{f}_i^l, y_i^l, \mathbf{p}_i^l, \phi, \omega_m)$;
  end
  $L_{rot} \leftarrow \sum_{i=1}^n loss\_rot(\mathbf{x}_i^l, \theta, \omega_r) + \sum_{i=1}^m loss\_rot(\mathbf{x}_i^u, \theta, \omega_r)$;
  $L \leftarrow \frac{L_{ce}+L_{cm}^l}{n} + \frac{L_{rot}}{n+m}$; Use SGD to update network parameters θ, ϕ, ω_c, ω_m, ω_r;
until *network parameters get converged*;

Algorithm 6: Algorithm for training network in the second stage.

Input: Training dataset, $\mathbb{D}^l = \left\{(\mathbf{x}_i^l, y_i^l)\right\}_{i=1}^N$, $\mathbb{D}^u = \{\mathbf{x}_i^u\}_{i=1}^M$.
Output: Optimized network parameters, θ, ϕ, ω_c, ω_m, ω_r.
repeat
  if *do_OOD_filtering* **then**
    $\mathbb{D}^u \leftarrow sample\_filtering(\mathbb{D}^u, \theta, \phi, \omega_c, \omega_m)$
  Fetch a batch of labeled samples $\mathbb{B}^l = \{(\mathbf{x}_i^l, y_i^l)\}_{i=1}^n$ and a batch of unlabeled samples $\mathbb{B}^u = \{\mathbf{x}_i^u\}_{i=1}^m$;
  $L_{ce} \leftarrow 0$; $L_{cm}^l \leftarrow 0$; $L_{cm}^u \leftarrow 0$; $L_{cc} \leftarrow 0$;
  for $i \leftarrow 1$ **to** n **do**
    $L_{ce}^i, L_{cc}^i, \mathbf{f}_i^l, \mathbf{p}_i^l \leftarrow loss\_ce\_cc(\mathbf{x}_i^l, y_i^l, \theta, \omega_c)$;
    $L_{ce} \leftarrow L_{ce} + L_{ce}^i$; $L_{cc} \leftarrow L_{cc} + L_{cc}^i$;
    $L_{cm}^l \leftarrow L_{cm}^l + loss\_cm\_labeled(\mathbf{f}_i^l, y_i^l, \mathbf{p}_i^l, \phi, \omega_m)$;
  end
  for $i \leftarrow 1$ **to** m **do**
    $L_{cc}^i, \mathbf{f}_i^u, \mathbf{p}_i^u \leftarrow loss\_cc(\mathbf{x}_i^u, \theta, \omega_c)$;
    $L_{cc} \leftarrow L_{cc} + L_{cc}^i$;
    $L_{cm}^u \leftarrow L_{cm}^u + loss\_cm\_unlabeled(\mathbf{f}_i^u, \mathbf{p}_i^u, \phi, \omega_m)$;
  end
  $L_{rot} \leftarrow \sum_{i=1}^n loss\_rot(\mathbf{x}_i^l, \theta, \omega_r) + \sum_{i=1}^m loss\_rot(\mathbf{x}_i^u, \theta, \omega_r)$;
  $L \leftarrow \frac{L_{ce}+L_{cm}^l}{n} + \frac{L_{cm}^u}{m} + \frac{L_{cc}+L_{rot}}{n+m}$;
  Use SGD to update network parameters θ, ϕ, ω_c, ω_m, ω_r;
until *network parameters get converged*;